Submitted to the ICAA'98 Workshop on Agent-Based Manufacturing

The AARIA Agent Architecture: From Manufacturing Requirements-to Agent-Based System Design

Van Parunak (Industrial Technology Institute), Albert Baker (University of Cincinnati), and Steve Clark (Industrial Technology Institute)

Abstract

Designs for real-world agent-based systems must reflect both domain requirements and technical capabilities. We illustrate requirements-based agent system engineering with a case study of AARIA (Autonomous Agents for Rock Island Arsenal), an industrial-strength agent-based shop-floor control and scheduling architecture being developed for an Army manufacturing facility.

1. Introduction

Successful application agents (as of any technology) must reconcile two perspectives. The researcher focuses on a particular agent capability (e.g., communication, planning, learning), and seeks practical problems to demonstrate the usefulness of this capability (and justify further funding). The industrial practitioner has a practical problem to solve, and cares much more about the effectiveness, cost, and speed of the solution than about its elegance or sophistication.

Industry's problem orientation leads strong emphasis on a system's life cycle. A typical life cycle includes requirements analysis, design, implementation and deployment, operation, logistics and maintenance, and decommissioning. The life cycle provides a road map that relates technology to its industrial context at each step. Any industrial activity follows such a pattern, whether it be building a product, putting in place the process for making a product, supplying a service, or creating a piece of infrastructure [12].

The early phases of the life cycle increase the likelihood that a system will indeed address the problems that justify it in the first place. This paper illustrates the reasoning that is involved by developing a set of requirements for manufacturing shop-floor scheduling and control, then displaying the design of an agent-based system and showing how it responds to those requirements. Further details on our design methodology are available in [14].

Section 2 briefly describes the problem we are addressing and the requirements we identify. Section 3 shows how these requirements justify the use of agents (as opposed to monolithic software and object-oriented programming), and defines a set of agents that satisfy the requirements. Section 4 illustrates the interactions of the various agents in the context of an example. Section 5 offers some conclusions.

2. System Requirements

Requirements Definition defines the problems that the project must solve. The focus is on *why* an effort is needed in the first place, not on *what* the project will do or *how* it will do it. This section outlines the problem we are addressing, then discusses requirements in two categories: those imposed by the system's interface with its external environment, and those needed in its internal operations to achieve necessary functionality. Table 1 summarizes the requirements we derive.

Table 1: Summary of Requirements

Requirement	Brief Description
Least Commitment	The customer's statement of demand (product specification, quantity, price, and delivery time) develops interactively, rather than being specified in detail at the outset.
Empowerment	Human stakeholders (including operators, manufacturing engineers, and managers) receive the information (both as-is and what-if) they need to do their jobs, with interfaces to let them control the system rather than being controlled by it.
Frequent Change	The system adapts its behavior dynamically in response to environmental changes
MRP Functionality	The aggregate behavior of the agent community subsumes functionality currently provided by MRP II systems.
Metamorphosis	The system maintains continuity between different entities that represent different stages in a common life cycle (for example, an order for a part, the part itself, and its production history).
Modality Emergence	An entity's factory control modality emerges dynamically from its operation in the context of the rest of the system, rather than being hard-coded or the result of an explicit inference.
Uniformity	An operation at the system boundary interacts with external Suppliers or Customers in the same way that it does with internal ones.

2.1 Shop-Floor Scheduling and Control

A manufacturing enterprise is measured by the cost of the goods it produces, their quality, and the timing of their availability relative to the customer's need. The task of shop-floor scheduling and control is to deploy resources to produce high-quality goods as inexpensively as possible and when the customer wants them. It governs decisions such as when which machines should be used for which products, what order products should be manufactured, when new jobs should be started, what level of inventory should be carried, and when machine maintenance should be performed. The problem is the subject of extensive research in the industrial engineering and operations research community. Because of its commercial potential, researchers in agent-based systems find it increasingly attractive as an application problem [2].

These specific requirements were identified on the basis of detailed discussions with manufacturing staff at Rock Island Arsenal and at various commercial manufacturing facilities.

2.2 External Interfaces

A shop-floor system needs to support relationships with four groups: customers who form downstream product-flow links; suppliers who form upstream product-flow links; operators and others who make the system work; and manufacturing engineers and management responsible for designing, installing, modifying, and maintaining the factory.

Support of Customer/Supplier Interaction.—As summarized in Table 2, each order that a customer places is a point in a space of at least five dimensions. These dimensions can interact in two ways. Sometimes the dimensions of a single order interact (as in a dependency between quantity and price). Different orders in the series can also interact with one another (for example,

a required inter-arrival rate). Uncertainty about these interactions makes both the customer's demand and the supplier's capability fuzzy along each of these axes.

Table 2: Some Dimensions of Customer-Supplier Interaction

Dimension	Definition	Fuzziness
Product Identity	What does the customer want?	Customer asks for a red car, but would be happy with any primary color; wants 3σ quality but would buy 6σ at marginally higher cost
Quantity	How many does he want?	Customer wants 15, but might buy 20 to get a price break
Price	How much is he willing to pay?	On cost-plus pricing, such as many DoD contracts, the exact price is not known until the product is delivered
Delivery Time	How long will he wait for product?	Tardy delivery is not uncommon. Also, a customer who initially asks for 100 by Monday might accept 10 by Monday and 90 by Friday.
Information Time	How long will he wait for these answers before ordering?	A customer with a custom order who learns that a bid will require a week to prepare might be happy to pick a preconfigured set of options that can be priced out of a catalog on the spot.

A request for quote (RFQ) seldom deals explicitly with all five dimensions, and in fact some may not be known in advance. The supplier and the customer must work jointly to identify a region within this space that satisfies both the customer's requirements and the supplier's capabilities. Unless the two parties repeat the same transactions frequently, it is counterproductive for one to define a small subregion in the demand space unilaterally. A better approach is a *least-commitment*, successive refinement one that starts with the entire demand space and incrementally shrinks it to discover a subregion that is acceptable to both parties.

Support of Operational Personnel and Equipment.—Modern management approaches *empower* operational personnel (the people who actually make the manufacturing system work), by providing them with the information they need, including both actual status information and "what-if" predictions based on suggested decisions, and by enabling them to control their interaction with the system (rather than being controlled by it).

Support of Manufacturing Engineers and Management.—Manufacturing engineers are responsible for designing, installing, modifying, and maintaining the manufacturing system. They routinely use emulation to evaluate and select the different alternatives. If the same code is used in both emulation and operation, actual shop-floor resources can be mixed with potential resources in emulations, supporting "what-if" evaluation of various options. This functionality extends *empowerment* from floor workers to management.

2.3 Internal Operations

Frequent Change.—The first constant of modern manufacturing systems is that nothing is constant. The products required of a facility, the technologies that produce them, and the materials from which they are fabricated change with an ever-increasing frequency. A system that can organize itself in response to *frequent change* will be more useful than one that must be explicitly reengineered to accommodate changes.

MRP Functionality.—The current paradigm of manufacturing scheduling is dominated by a widely-implemented approach called MRP II, "Manufacturing Resource Planning," which builds a

schedule by working from the product's bill of materials. In most firms that use any scheduling at all, MRP II is the backbone that supports other business functions. The chance of acceptance for any new paradigm will be greatly increased if it subsumes standard *MRP II functionality.*

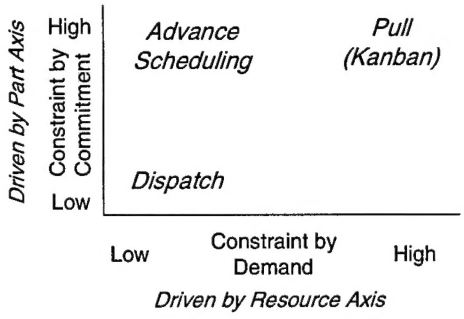

Figure 1: Factory Control Modalities

Different Factory Control Modalities.—No operation can execute before its input parts and its resources (such as machines, tools, and operators) are available, but an operation may be delayed even further because of low immediate Demand for its output, or because its immediate execution would violate existing Commitments. These two constraints define a two-dimensional space that characterizes traditional factory control modalities (Figure 1). For example, Demand may be limited by permitting a process to operate only when there is space in its output buffer, while Commitment is commonly reflected in some prediction of what the operation will do when.

Figure 1 shows where three common approaches to scheduling fall in this space. In these approaches, all elements of a shop are restricted to the same region of Figure 1, and in particular to one of the three restricted points. It is more desirable that (1) points in the space intermediate between the classical three points are accessible; (2) different processes can operate at different places in this space, and (3) processes and machines can migrate through this space and thus change modality as circumstances change. We call this requirement *modality emergence.*

Metamorphosis.—Any system that supports commitments must be able to represent entities both as they are and as they might be in the future. The software representing the entity must *metamorphose* in the course of time. For example, an order eventually becomes a part. The corresponding software artifact starts off as a statement of what needs to be done to produce the part. It either dies or evolves into an increasingly specific description of the actions that actually take place. When the part is complete, the factory maintains a log of its production to support liability defense, recall operations, or (particularly in the military) orders for custom spare parts.

Uniform Internal/External Presentation.—Any manufacturing system must interface smoothly with customers (who order and consume its output) and suppliers (who provide the raw material for that output). We want the model of a customer or supplier to have a *uniform* interface whether dealing with an external entity that interfaces with the whole shop or a single resource in the shop, for three reasons. First, this arrangement is simpler than having distinct mechanisms for interaction. Second, by having a uniform interface, the system can scale more easily. We can create a factory using a small number of machines or a large number of machines. The system can dynamically create a "virtual" factory of machines in different locations, belonging to different owners. A single machine can buy and sell its capabilities on the open market. Third, outsourcing is performed in a way that is transparent to the system's operation.

3. System Design

In designing a system to meet these requirements, we must first justify the use of agents as opposed to another software technology, then develop a specific set of agents and design behaviors that support the requirements.

3.1 Why Use Agents?

Table 3: Increasing Software Localization

	Monolithic Program	Structured Programming	Object-Oriented Programming	Agent-Oriented Programming
How does a unit behave? (Code)	External	Local	Local	Local
What does a unit do when it runs? (State)	External	External	Local	Local
When does a unit run?	External	External (called)	External (msg)	Local (rules; goals)

Table 3 shows the relation of agents to other common software technologies, as an extension of the historical trend toward increasing localization. For our purposes, an agent is an active object with initiative. Unlike other objects, it does not need to be invoked, but constantly monitors its local environment and acts autonomously based on its individual programming.

Several of the requirements point toward an implementation as an agent-based system. *Empowerment* requires that humans function as peer elements in the system rather than being run by the system, a requirement that is greatly supported if other parts of the system mirror their human agenthood. *Frequent change* is easier to accommodate with agents. Because an agent invokes itself, it can be modified without the need to modify or even notify other components. They in turn will discover the effect of the change on their own initiative and adapt to it. *Metamorphosis* is easier to implement if the entities that change through time are agents, since each can manage its own life cycle. *Uniformity* requires that the elements of the system interact with one another as they do with external entities, and since external entities by definition have separate threads of control, internal elements should as well.

3.2 What Gets Agented?

Previous research on agent-based factory control and scheduling (including our own) differs widely on what is represented as an agent: levels in a hierarchical decomposition of the factory [3, 9, 16], Resources [1, 6, 13, 15], or Parts [4, 8]. In selecting our agents, we want to begin with the broadest possible set of candidates. While some entities may prove unnecessary, it's easier to cast the net broadly and leave some as stubs than to build an architecture into which omitted entities cannot easily be added later.

Our design methodology, outlined in more detail in [14], draws on an apparently universal characteristic of human thought: people partition reality between things and events that involve those things, and express thoughts by placing nouns in various semantic relations with a verb. The underlying set of these relations (called "deep structure cases" or simply "cases") appears to be the same in every language. As described in [10], we identify candidate agents by constructing a set of declarative sentences describing the domain (for example, "Joe oversaw Unit Process 12 on

Part 25 from Acme Supplies, using Mill 32, Cutter 86, and Part Program 19, producing Part 26 for US Army Order 22." The nouns in such a sentence are candidate agent instances, and their cases represent candidate agent classes. In our domain, we identify the following agent classes:

Unit Process ("Unit Process 12").—We abstract the Unit Process away from the Resources that are used to perform it, since much of scheduling consists of identifying alternative Resources that can be used to perform what we would like to consider the same underlying Unit Process [5]. An instantiation of a Unit Process in space and time (e.g., the specific instance of heat treating performed in Oven 18 at 14:32 3 May 1993) is an Operation.

Resource ("Mill 32," "Cutter 86," "Part Program 19").—The "tools" that are needed to perform an Operation (an instance of the Unit Process), including machines, material handling devices, energy, tooling, fixtures, gauges, part programs, and documentation. Resources are characterized by such things as maintenance schedules, availability, and cost of use. Those aspects of the machine operator that are required to complete a unit process are best modeled as a Resource as well. Other aspects of humans are covered in the Manager category.

Manager ("Joe").—The human responsible for the Operation. In an automated factory, it becomes the plant manager. Automated representatives watch for things like chaos, performance metrics, energy consumption, and cash flow.

Part ("Part 25," "Part 26").—The inputs (Materials) and outputs (Products) for a Unit Process. There may be more than one input (e.g., assembly) or output (e.g., sawing up bar stock). Between Unit Processes, Parts undergo material handling operations such as transport and storage. Theoretically, Material Handling is just a Unit Process that changes a part's age and location. However, these changes do not alter the part functionally, and the part number does not change across a material handling operation (as it does across other Unit Processes). Thus we make Parts responsible for their own material handling, and permit them to acquire necessary Resources just as Unit Processes do in order to move from one Unit Process to another. This generalization of a Part (which at any moment may or may not actually contain a part) may be called a Buffer.

Customer ("US Army Order 22").—The linguistic Beneficiary; the one who benefits from the execution of the work. The Customer represents a single purchase decision, or order.

Supplier ("Acme Supplies").--Another variety of Beneficiary, this time the one from whom the input material is purchased.

The basic dynamic of manufacturing is that Parts move through a network of Unit Processes and Buffers. Each Unit Process acquires one or more Input Parts (Materials) from Buffers responsible for Parts of the necessary types, and engages certain Resources to produce one or more Output Parts (Products) into Buffers of the appropriate types. Figure 2 outlines some of these relationships, emphasizing that the dynamics of manufacturing is the flow of both Parts and

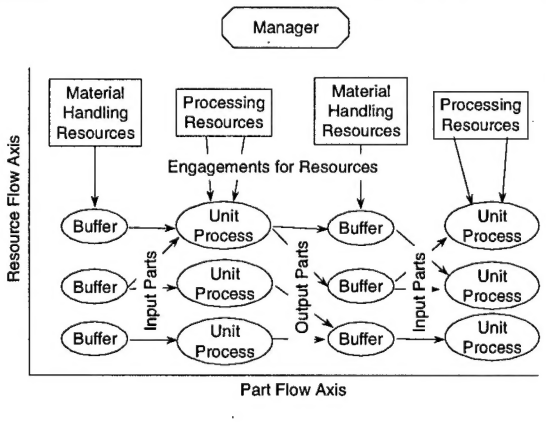

Figure 2: Operations as the Nucleus of Manufacturing

engagements for Resources through Unit Processes. The Manager oversees all entities and intervenes in their activities as required. Scheduling manages the flow of Resource engagements and Parts through various Unit Processes so that they coincide with each other.

3.3 Refinement

The Requirements developed earlier enable us to refine this preliminary set of agents.

Implications of Uniformity.—Our initial ontology distinguishes Unit Processes, Suppliers, and Customers. However, due to the Uniformity principle, an internal Unit Process and an external Customer draw from a Buffer in exactly the same way, and an internal Unit Process and an external Supplier feed a Buffer in exactly the same way. Uniformity leads Customer and Supplier to become roles that are played by Unit Processes at the boundary of the system. We will use the terms "Customer" and "Supplier" to refer to any agent playing the respective role.

AARIA's approach to Uniformity supports a novel approach to supply chain integration. Traditionally, supply chains are integrated by visualizing each firm as a monolithic entity with its own internal mechanisms, and constructing a special set of mechanisms to permit separate firms to interact with each other. AARIA instead decomposes each firm itself into a miniature supply chain made up of a series of producers and consumers. As a result, the interfaces between AARIA agents within a firm are the same as those between one firm and another, and integration of AARIA-based firms into a larger supply chain is immediate and transparent.

Implications of Metamorphosis and Least-Commitment.—Metamorphosis recognizes that entities that change over time should be modeled in a way that lets us maintain multiple copies to capture "as-is" and "to-be" information. We satisfy this requirement by distinguishing between *persistent agents,* whose behavior does not change over the time scale involved in daily shop operation, and *transient agents,* which represent interactions among persistent agents and which go through a well-defined life cycle of behaviors as the shop operates.

The three persistent agents that interact to generate transient agents are Parts, Resources, and Unit Processes. A *Part* persistent agent is not a physical part (which changes considerably as it is manufactured), but the representation of a particular part type that maintains the information about that type, owns its inventory, keeps its production history, and on the basis of that history can forecast future production capability. Up to this point we have referred to this entity as a Buffer. From the perspective of a marketplace, a Buffer functions as a broker for physical parts of its type, and for clarity and consistency with other persistent agents we will refer to it as a Part Broker. A persistent *Resource* (or Resource Broker) represents an individual physical resource (such as a given lathe, cutting bit, or operator) and maintains the history and schedule of that Resource's commitments. A persistent *Unit Process* (or Unit Process Broker) knows what Parts and Resources are needed to execute a specific unit process with specific output Parts. It is responsible for marshaling the inputs and resources needed to execute a unit process.

Interactions among these three persistent agents are modeled as transient agents: Engagements, Materials, Products, and Operations. Engagements, Materials, and Products model interactions between a Unit Process Broker and some other agent, while the Operation captures the transient aspects of the Unit Process itself. Each transient agent has a six-phase life cycle, as summarized in Table 4: Inquiring, Committing, Committed, Available, Active, and Archived.

In their Archived phase, transient agents are stored with the appropriate persistent agent as part of the system log. Resource Brokers absorb Archived Engagements that have executed on their respective resources; Part Brokers absorb Archived Products of their type and Archived Materials of the input parts that went into them; and the Unit Process Broker maintains its own Archived Operations. Archived Engagements, Materials, and Products all reference the Archived Operations that involved them, and Archived Operations in turn reference their Engagements, Materials, and Products, so that other agents can retrieve historical information as needed. In some cases, a persistent agent may cache information about Archived agents that are logged on another persistent agent, but the "official" copy is always the full Archived agent.

Implications of Modality Emergence.—Figure 1 suggests that traditional scheduling modalities are restricted points in a broader space defined by two axes: constraint by commitment and by demand. AARIA includes mechanisms that permit subsets of the factory to occupy any position in this space, and to change their modality as circumstances change.

The CASCADE system [13] demonstrates one mechanism for modality emergence along the part flow axis. The Pull and Dispatch modalities are two poles in the behavior of a Part Buffer with minimum and maximum capacities. Below, the minimum, the buffer pulls from its Supplier. Above the maximum, it forces its Customer to consume a part. For a given set of capacities, the buffer automatically shifts from Pull to Dispatch as its moment-by-moment load changes. This mechanism provides continuous adjustment between low and high constraint by demand.

The original CASCADE model assumes that the population level that triggers pulling (the "minimum") is less than the level that triggers spilling (the "maximum"). This assumption may not always be justified, so we refer simply to the Material trigger capacity and the Product trigger capacity. When the Part Broker's population exceeds the Product trigger capacity, it spawns an Inquiring Material to seek for possible customers. When its population falls below the Material trigger capacity, it spawns an Inquiring Product to search for suppliers.

Table 4: Life Cycle of a Transient Agent

Phase	Conventional Parallel	Behavior	Transition to Next
Inquiring	RFQ	Refine terms between Persistent Agents (demand function for part; engagement function for resource)	Acceptance of current state of --demand function by Customer for Material or Part Broker for Product --engagement function by Operation
Committing	Offered Order	Agree to current state of demand or engagement function	All relevant parties confirm acceptance of terms
Committed	Confirmed Order	Agreements confirmed. Notify losing bidders.	Arrival of commitment time and physical availability of part or resource
Available	WIP	Be physically ready for the operation	Initiation of Operation by Unit Process or Part Broker
Active	WIP	Operation is underway	Completion of Operation
Archived	Production Log	Log of what happened and who participated	<<Permanent>>

AARIA's mechanism for modality emergence along the commitment dimension of the space of modalities is in the resource axis. An engagement is not irrevocably scheduled to a fixed time in the future, but (in keeping with the customer's requirements) can be moved within certain bounds by the resources that it has reserved. A resource that is largely uncommitted has more flexibility in moving engagements in time than one that is heavily loaded. That is, heavily loaded resources tend toward a linear internal ordering of engagements (a fixed advance schedule), while lightly loaded resources can select among alternate engagements, down to the moment of activation, so that they effectively dispatch. Thus behavior along the commitment dimension can vary not only from resource to resource, but also from one time period in a resource's future to another, as that resource's level of utilization varies.

Implications of Frequent Change.—Because each agent has its own thread of control and its own goals, a community of agents can discover changes among its members or in its external context without explicit reprogramming. While this autonomy enables the system to reconfigure and adjust itself, it also raises the expectation (in accordance with the Second Law of Thermodynamics) that it might become increasingly disordered over time.

Our design approach is heavily influenced by principles discerned in naturally occurring agent-based systems [11]. Such systems deal with the Second Law by coupling the macro behavior of agents to a low-level dissipative mechanism, such as pheromone evaporation [7]. This dissipative mechanism both generates a flow field to which agents can orient themselves and provides an "entropy leak" to drain second-law entropy gain from the macro to the micro level.

One example of such a dissipative mechanism is the flow of currency in a market economy. AARIA institutes artificial markets among the various agents, extending the mechanisms explored in [1]. In addition to enabling self-organization among agents, this economic model provides a natural way to measure and trade off the relative importance of conflicting demands at each decision point, just as a market-based price system measures the relative values of goods that would be difficult to compare in a barter system. To facilitate interfacing with the outside world (as Uniformity requires), AARIA's internal currency is denominated in dollars.

Implications of Empowerment.— Humans interact with the system through Manager and Resource agents. Each Manager is a watchdog for some aspect of system performance (such as cash flow, or chaotic behavior, or equipment utilization), and interacts with other agents by cash grants and taxes that adjust the underlying dynamics of the system and coax it into new behaviors. Operators are a subclass of Resource that represent the operators' needs and wishes.

Implications of MRP Functionality.—MRP functionality emerges naturally from the community of agents that we have constructed to satisfy the other requirements. The flow of bids along the part axis mirrors the bill-of-materials decomposition of the product that is the core of MRP. Support of the different scheduling modalities provides rich support for planning and finite-capacity scheduling. Uniformity means that order entry and purchasing are modeled at every operation, and thus are supported naturally at the limits of the firm as well. The use of currency flow to provide a dissipative field for self-organization makes each agent a profit center and naturally supports integration of financial functions.

4. An Example of AARIA in Action

Figure 3 illustrates a typical AARIA transaction, beginning with a customer request. Other agents may also initiate transactions. We focus on the life cycle of the Material-1 agent. Other transient agents go through the same phases, but not completely in sync.

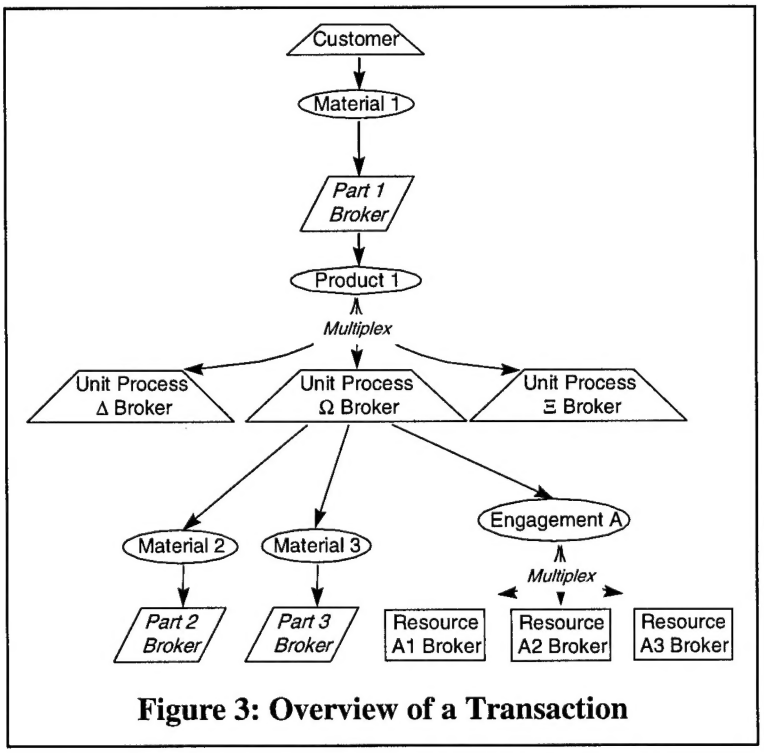

Figure 3: Overview of a Transaction

The Customer submits an initial RFQ in the form of an *Inquiring Material*. Typically this agent specifies only Product Identity, Quantity, and Delivery Time, and even these may be imprecise. When the Inquiring Material is first submitted, AARIA gives the Customer a quick "worst-case" bid, and then as it analyzes the problem, refines the bid. At any point, the Customer can accept the terms of the current bid. Typically, the price quoted for an order initially drops as the chain of suppliers refines its bid, then begin to increase as the desired due date approaches and resources reflect their increasing loading in higher prices.

The Customer accepts the current terms of Material-1 by authorizing production, thus entering *Committing*. At this point Material-1 moves from Inquiring to *Committed*, and when the Customer receives the commitment, it too becomes *Committed*. If Part 1 Broker is able to supply Material-1 from inventory, Material-1 goes immediately from *Committed* to *Available*, and is delivered to the Customer. Otherwise the Customer's commitment propagates from Material-1 to other transient agents to arrange for its production. In this case, Material-1 remains *Committed* throughout the *Committed*, *Available*, and *Active* phases of other transient agents, including the execution of the Operation that produces the part. During this period, a Unit Process Broker may need to accommodate Customer change requests or engineering change orders, or deal with upstream failures, and may be able to achieve local efficiencies, such as clustering similar parts from different jobs in order to reduce setup times.

When Product-1 goes *Available*, it is delivered to Part 1 Broker and becomes *Archived*, and the physical part is assigned to Material-1, which then moves to *Available* at the committed time and is delivered to the Customer. The Customer pays for the part, and the revenue derived from the job is distributed through the system to the cost centers that have earned it. This flow of revenue is a dissipative field-generating activity that permits self-organization of manufacturing agents. The revenue thus obtained by agents is traded among them in other phases.

Though this example is initiated by a Unit Process Broker (a Customer) by way of a Material agent, any persistent agent can initiate action, under appropriate circumstances.

- A Unit Process Broker may initiate activity through a Product agent instead of a Material agent, thus signaling the Part Broker for its output that it would like to operate. Such an action might result from an advance schedule that calls for the operation to run at a given time, and would be plausible in a make-to-stock operation.
- A Part Broker will spawn a Product agent if its inventory falls below a specified value, thus imposing a pull dynamic on its Suppliers. It should also be able to spawn a Material agent if its inventory rises above a certain level, thus actively seeking out consumers for its stock. If the price it offers for excess inventory is low enough, it may be able to find Unit Process Brokers willing to consume from it.
- A Resource Broker may also initiate processing by spawning an Engagement with a selected operation. This mode of operation may characterize a bottleneck resource, which thus offers services on the basis of "Don't call me; I'll call you."

5. Conclusions

Industrial users are driven not by the technical curiosity that motivates the academy, but by the need to solve specific problems. To help insure that systems solve the problems that they address, they are developed in the context of a life cycle that begins with careful specification of the requirements that the final system must satisfy. We have identified seven such requirements in the domain of shop floor control and scheduling. These requirements clearly point toward an agent-based approach to this problem, and provide guidance in developing the behavioral characteristics of the individual agents. These characteristics, while interesting in themselves, can only be justified in an industrial setting by correlating them with the requirements that they satisfy. The thought process outlined in this paper is crucial to enlisting continued industrial support to move agent technologies from the laboratory into real-world application.

Acknowledgment

Initial research on AARIA was funded by the DARPA Agile program under contract F33615-95-C-5524, managed by USAF ManTech. The prime contractor is Intelligent Automation, Inc. (Len Haynes, Kutluhan Erol, and Renato Levy). Other contractors are the Industrial Technology Institute (Van Parunak, Steve Clark, Jorge Goic), the University of Cincinnati (Albert Baker, Bradley Matthews, Ben Moore, Brian Birtle, Veena Pandiri), Flavors Technology (Howell Mitchell), and Rock Island Army Arsenal (Greg Peters, Don Tice).

References

[1] A. D. Baker. Metaphor or Reality: A Case Study Where Agents Bid With Actual Costs to Schedule a Factory. In S. H. Clearwater, Editor, *Market-Based Control: A Paradigm for Distributed Resource Allocation*, pages 184-223. World Scientific Publishing Co. Pte. Ltd., 1996.

[2] A. D. Baker. A Survey of Factory Control Algorithms which Can be Implemented in a Multi-Agent Heterarchy: Dispatching, Scheduling, and Pull. *Journal of Manufacturing Systems*, , Forthcoming. Available at http://www.ececs.uc.edu/~abaker/AgentAlgs.ps.

[3] J. Butler and H. Ohtsubo. ADDYMS: Architecture for Distributed DYnamic Manufacturing Scheduling. In A. Famili, D. S. Nau, and S. H. Kim, Editors, *Artificial Intelligence Applications in Manufacturing*, pages 199-214. AAAI Press/The MIT Press, Menlo Park, CA, 1992.

[4] N. A. Duffie, R. Chitturi, and J. I. Mou. Fault-tolerant Heterarchical Control of Heterogeneous Manufacturing System Entities. *Journal of Manufacturing Systems*, 7(4):315-28, 1988.

[5] I. Finnie, Editor. *Unit Manufacturing Processes: Issues and Opportunities in Research.* Washington, DC, National Academy Press, 1995.

[6] J. Heaton. Agent Architecture Distributes Decisions for the Agile Manufacturer: Reengineering at AlliedSignal Automotive Safety Restraint Systems. *AMR Report*, (June):8-13, 1994.

[7] P. N. Kugler and M. T. Turvey. *Information, Natural Law, and the Self-Assembly of Rhythmic Movement.* Lawrence Erlbaum, 1987.

[8] J. Maley. Managing the Flow of Intelligent Parts. *Robotics and Computer-Integrated Manufacturing*, 4(3/4):525-30, 1988.

[9] H. V. D. Parunak. Manufacturing Experience with the Contract Net. In M. N. Huhns, Editor, *Distributed Artificial Intelligence*, pages 285-310. Pitman, London, 1987.

[10] H. V. D. Parunak. Case Grammar: A Linguistic Tool for Engineering Agent-Based Systems. ITI Technical Memorandum, http://www.iti.org/~van/casegram.ps, Industrial Technology Institute, Ann Arbor, 1995.

[11] H. V. D. Parunak. 'Go to the Ant': Engineering Principles from Natural Agent Systems. *Annals of Operations Research*, (forthcoming), 1998. Available at http://www.iti.org/~van/gotoant.ps.

[12] H. V. D. Parunak. Industrial and Practical Applications of DAI. In G. Weiss, Editor, *Introduction to Distributed Artificial Intelligence*, MIT Press (forthcoming), 1998.

[13] H. V. D. Parunak, J. Kindrick, and B. Irish. Material Handling: A Conservative Domain for Neural Connectivity and Propagation. In *Proceedings of Sixth National Conference on Artificial Intelligence*, pages 307-311, American Association for Artificial Intelligence, 1987.

[14] H. V. D. Parunak, J. Sauter, and S. J. Clark. Toward the Specification and Design of Industrial Synthetic Ecosystems. In *Proceedings of Fourth International Workshop on Agent Theories, Architectures, and Languages (ATAL)*, Springer, 1997.

[15] M. J. Shaw and A. B. Whinston. Task Bidding and Distributed Planning in Flexible Manufacturing. In *Proceedings of IEEE Int. Conf. on AI Applications*, pages 184-89, 1985.

[16] K. J. Tilley and D. J. Williams. Modelling of Communications and Control in an Auction-based Manufacturing Control System. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 962-967, IEEE, 1992.

INTERNET DOCUMENT INFORMATION FORM

A . Report Title: The AARIA Agent Architecture: From Manufacturing Requirements-to Agent-Based System Design

B. DATE Report Downloaded From the Internet 4/05/99

C. Report's Point of Contact: (Name, Organization, Address, Office Symbol, & Ph #): **Industrial Technology Institute**
P.O. Box 1485
Ann Arbor, MI 48106

D. Currently Applicable Classification Level: Unclassified

E. Distribution Statement A: Approved for Public Release

F. The foregoing information was compiled and provided by:
DTIC-OCA, Initials: VM_ **Preparation Date**: 4/05/99__

The foregoing information should exactly correspond to the Title, Report Number, and the Date on the accompanying report document. If there are mismatches, or other questions, contact the above OCA Representative for resolution.